arc42

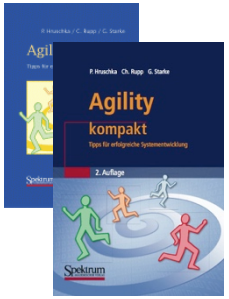
arc42 Reality Check

Praxiseinsatz, Werkzeuge, Add-Ons

Dr. Gernot Starke
innoQ Fellow

WJAX. November 2015

Follow @gernotstarke



Dr. Gernot Starke

innoQ Fellow

- ***Softwarearchitekturen***
  - Entwurf, Entwicklung, Management
  - Evolution & Modernisierung
  - Dokumentation
  - Training
- ***Mentoring und Coaching***
  - Analyse und Optimierung von Entwicklungsprozessen
- ***Reviews***

+49 177 – 728 2570
Gernot.Starke@innoQ.com

www.arc42.de

Was ist arc42?

Behälter für Architektur-Info

Anforderungen

Strukturen von Sourcecode

Entscheidungen

Warum?

http://www.flickr.com/photos/mistressf/737350025/sizes/l/

Detail

Gallery

Summary
Detail

View

By name
By source

Browse

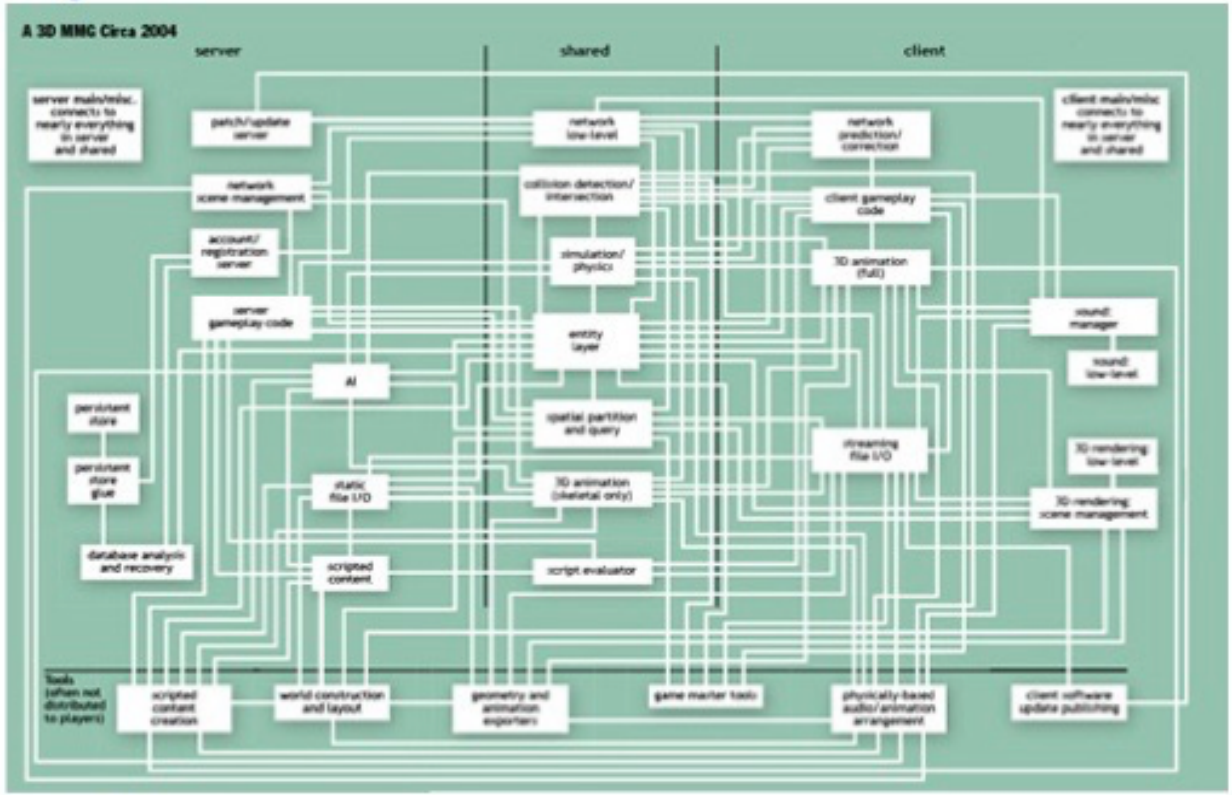
By domain
By view

Game 2004 MMORPG

Source

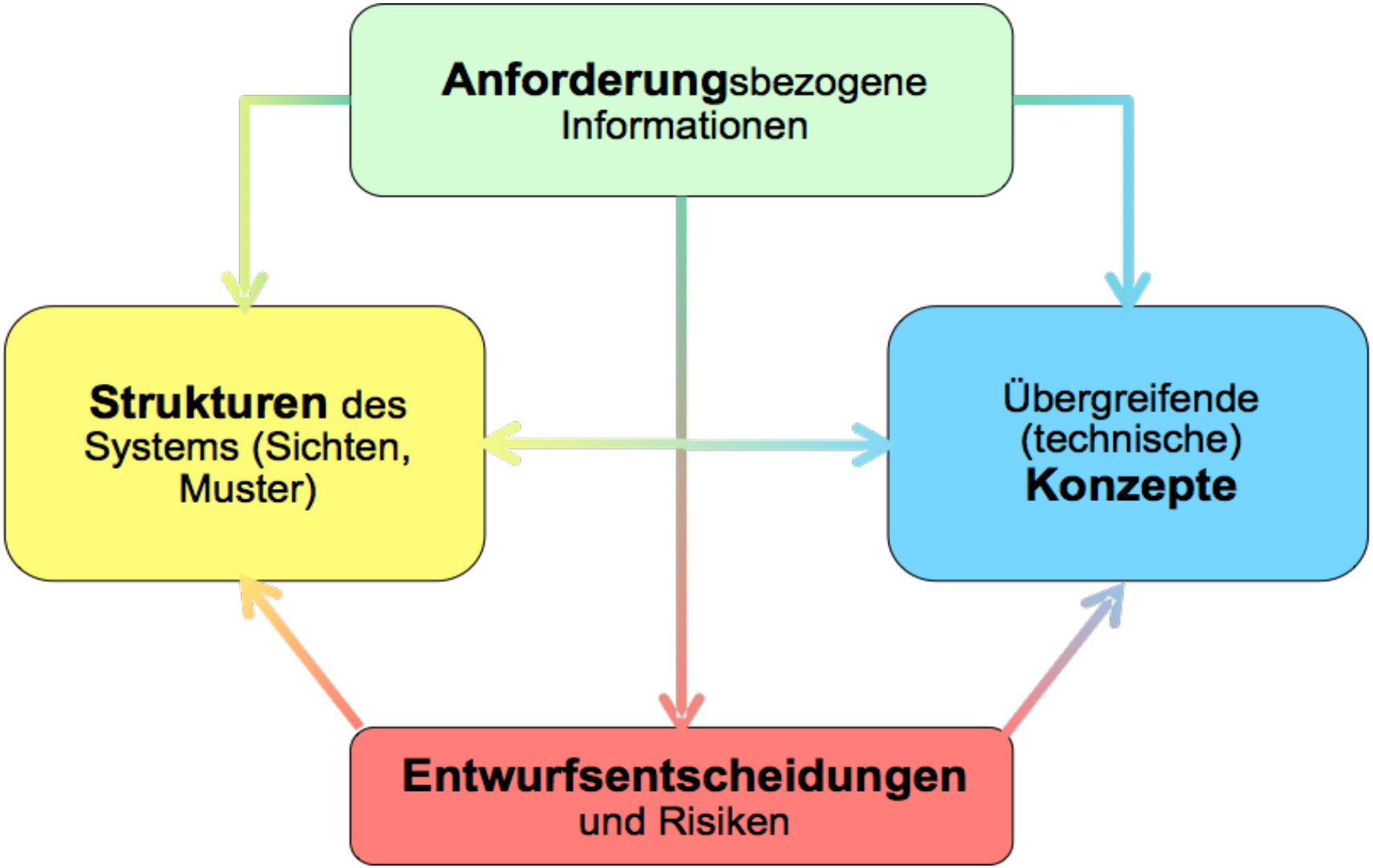
Blow, J. "Game Development: Harder Than You Think." *ACM Queue*, February 2004, p. 34.

Diagram



Wie sieht‘s aus?

Aufbau


Anforderungsbezogene Informationen
Strukturen des Systems (Sichten, Muster)
Übergreifende (technische) **Konzepte**
Entwurfsentscheidungen und Risiken

ARC42 Architecture Documentation

1. Introduction and Goals
2. Constraints
3. Context
4. Solution Strategy
5. Building Block View
6. Runtime View
7. Deployment View
8. Concepts
9. Architecture Decisions
10. Quality Scenarios
11. Risks
12. Glossary

Whitebox-Template

1. Name
2. Overview (Diagram!)
3. Motivation, Rationale
4. Contained Blackboxes
5. Internal Interfaces
6. Open issues

Blackbox-Template

1. Name
2. Purpose / Responsibility
3. Interfaces
4. Location / Files
5. Fulfilled Requirements
6. Variability / Flexibility
7. Open Issues

Structure of Architecture Decisions

1. What to decide?
   1.1 In what context?
2. How was decided?
   2.1 Why?
   2.2 Assumptions
   2.3 Discarded Alternatives
3. Consequences?
4. Known Risks?
5. Who has decided?

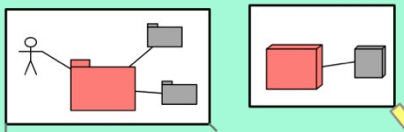
Scenarios to define Quality Requirements

1. Useage Scenarios
2. Change Scenarios

Structure of Concepts

1. Goals and Requirements
2. Constraints
3. Scope / Context
4. Solution / Approach
   4a Structures & Process
   4b Samples incl. Code
5. Alternatives
6. Risks

Context



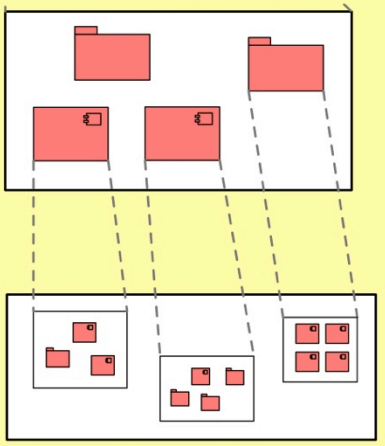
Quality Goals

Goal	Description
...	...

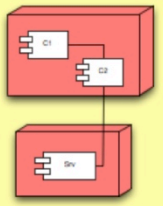
Stakeholder Table

Who?	Interest?
...	...

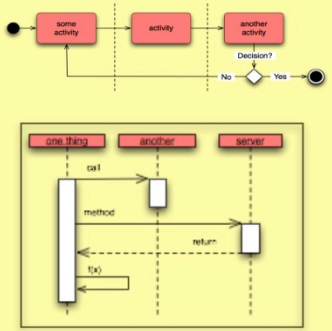
Building Block View



Deployment View



Runtime View



Structure of Interface Descriptions

1. Name, Version
2. Ressources (Syntax)
3. Semantic
   3.1 Business
   3.2 Technical
4. Protocol
   4.1 Flow / Process
   4.2 Transmission Channel
5. Error and Exception Behavior
6. Restrictions
7. Sample Data
8. Quality Attributes / QoS

arc42 V 6.0 (Dec. 2012),
BigPicture
created by Peter Hruschka
& Gernot Starke

Aufbau von arc42

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbedingungen
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
….

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
….

7.Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
….

8. Konzepte
8.1 Fachliche Struktur und Modelle
8.2 Typische Muster und Strukturen
8.3 Persistenz
8.4 Benutzeroberfläche
….

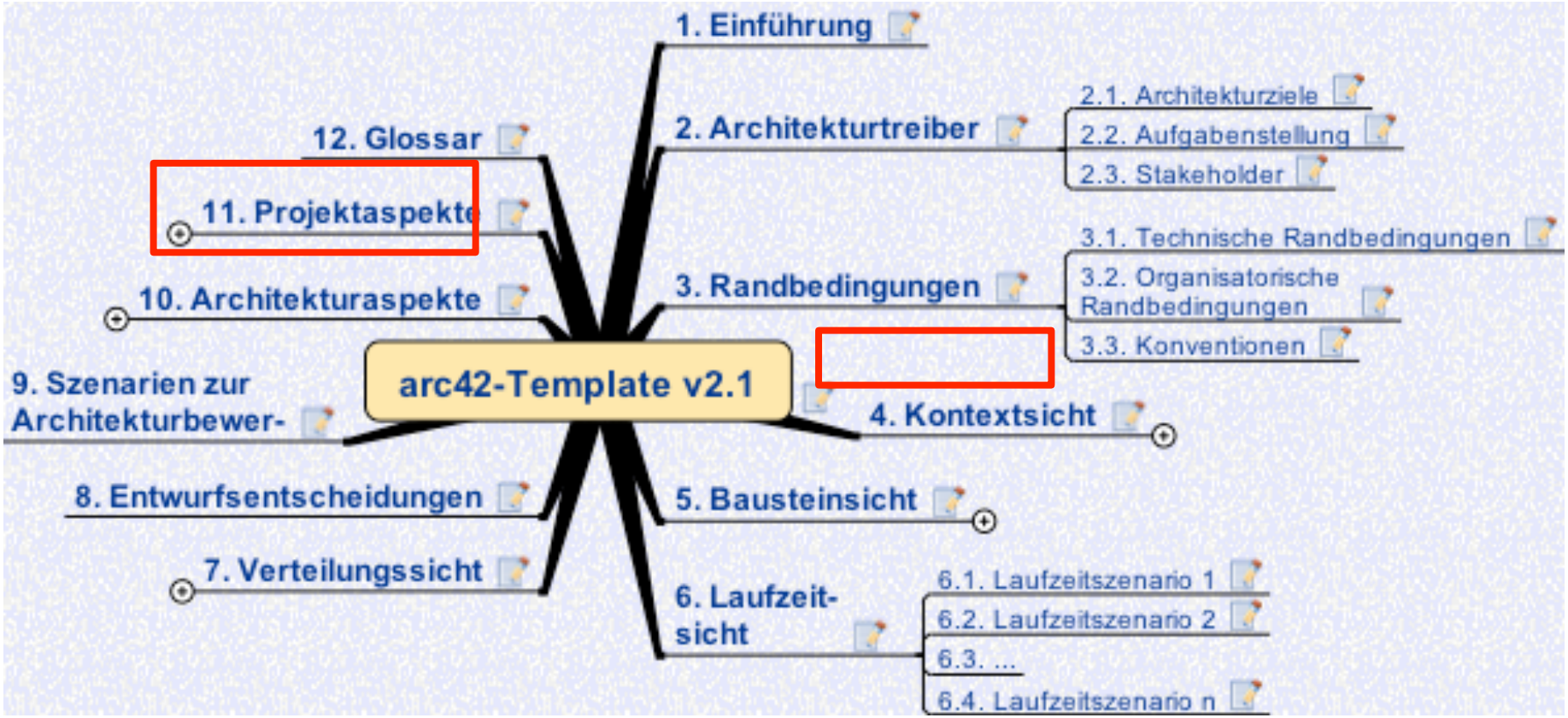
9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
….

10. Qualitätsszenarien
10.1 Qualitätsbaum
10.2 Qualitäts-/Bewertungsszenarien

11. Risiken

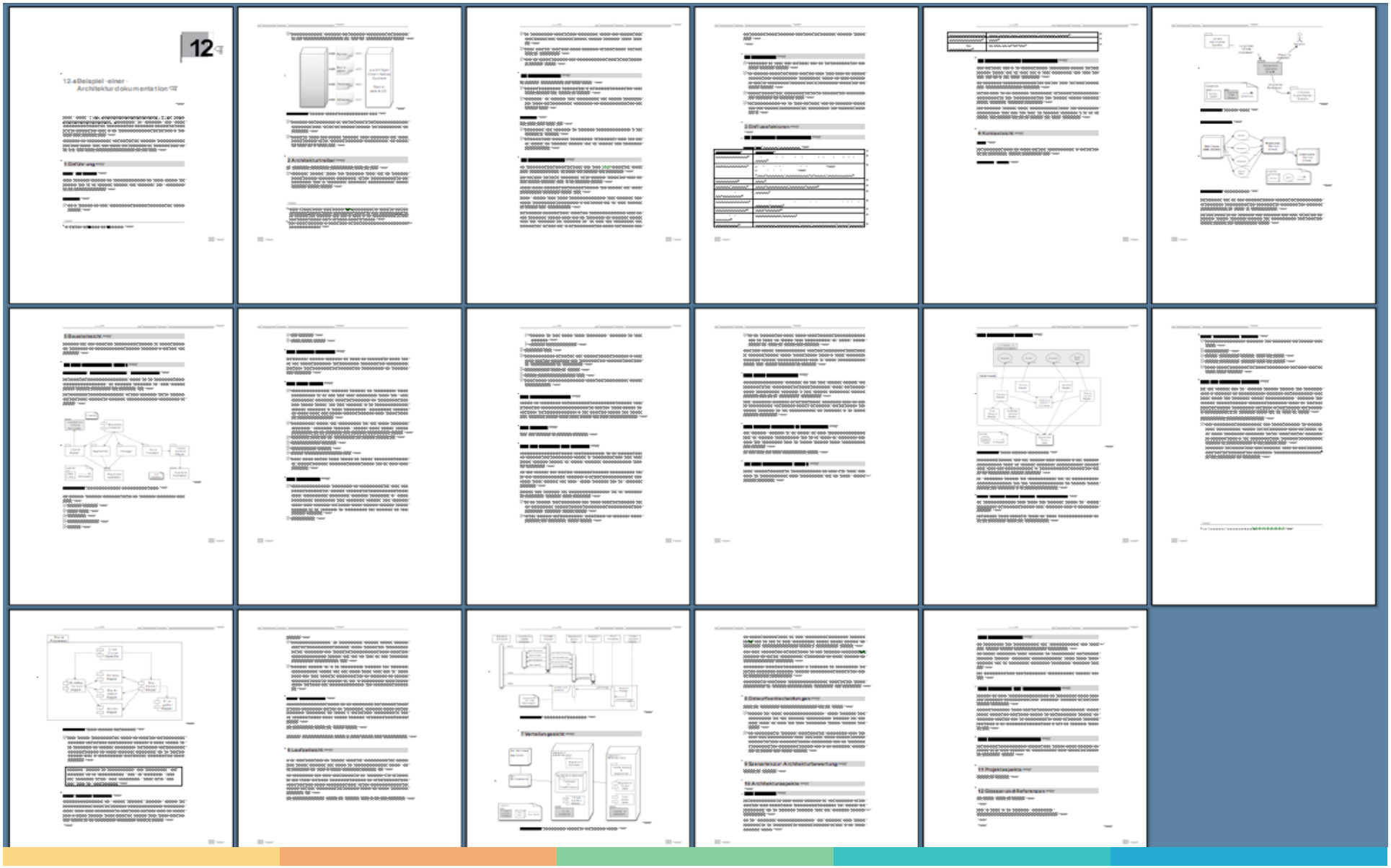
12. Glossar

2006...

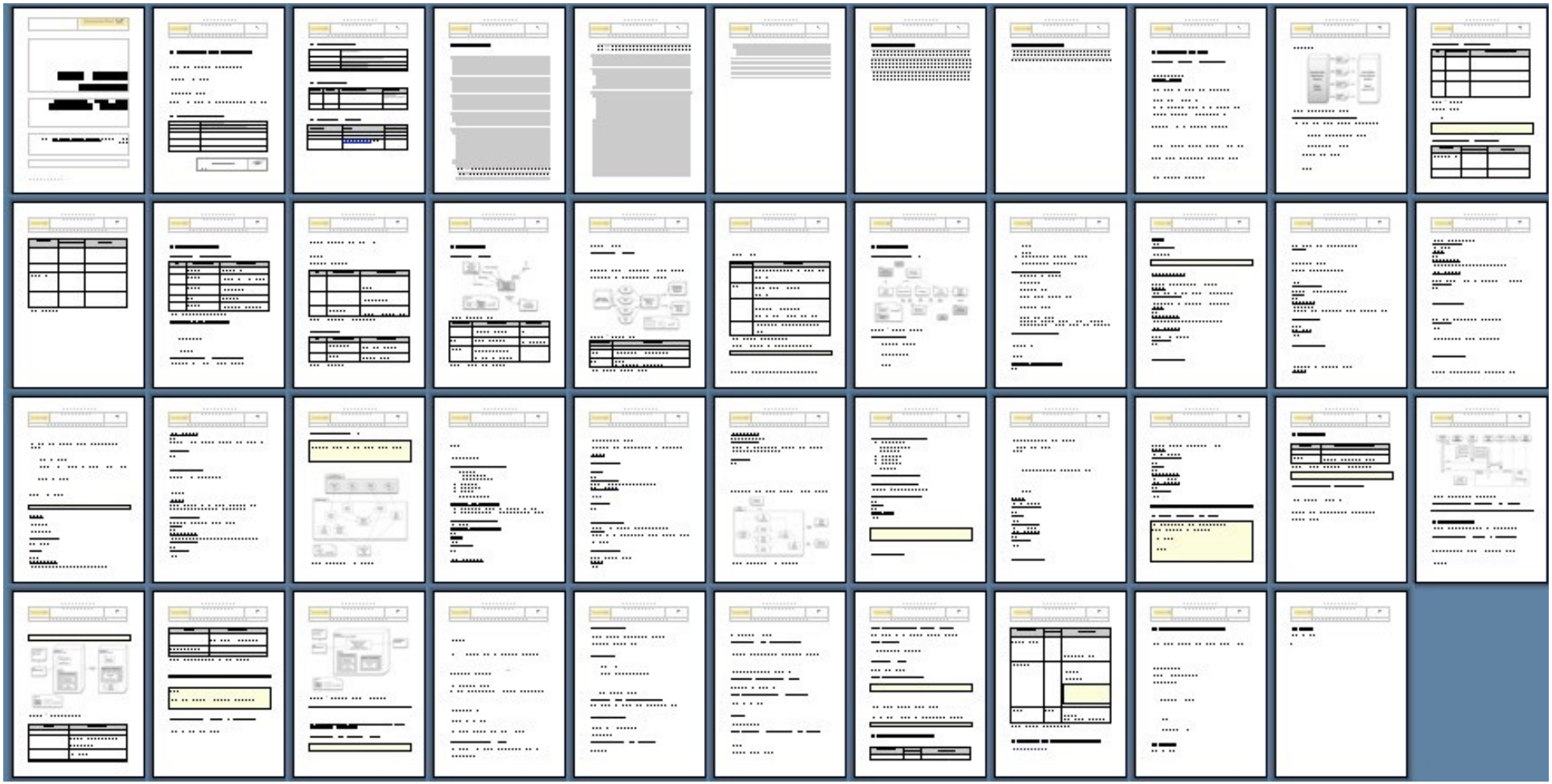

arc42-Template v2.1
1. Einführung
2. Architekturtreiber
2.1. Architekturziele
2.2. Aufgabenstellung
2.3. Stakeholder
3. Randbedingungen
3.1. Technische Randbedingungen
3.2. Organisatorische Randbedingungen
3.3. Konventionen
4. Kontextsicht
5. Bausteinsicht
6. Laufzeit-sicht
6.1. Laufzeitszenario 1
6.2. Laufzeitszenario 2
6.3. ...
6.4. Laufzeitszenario n
7. Verteilungssicht
8. Entwurfsentscheidungen
9. Szenarien zur Architekturbewer-
10. Architekturaspekte
11. Projektaspekte
12. Glossar

delta

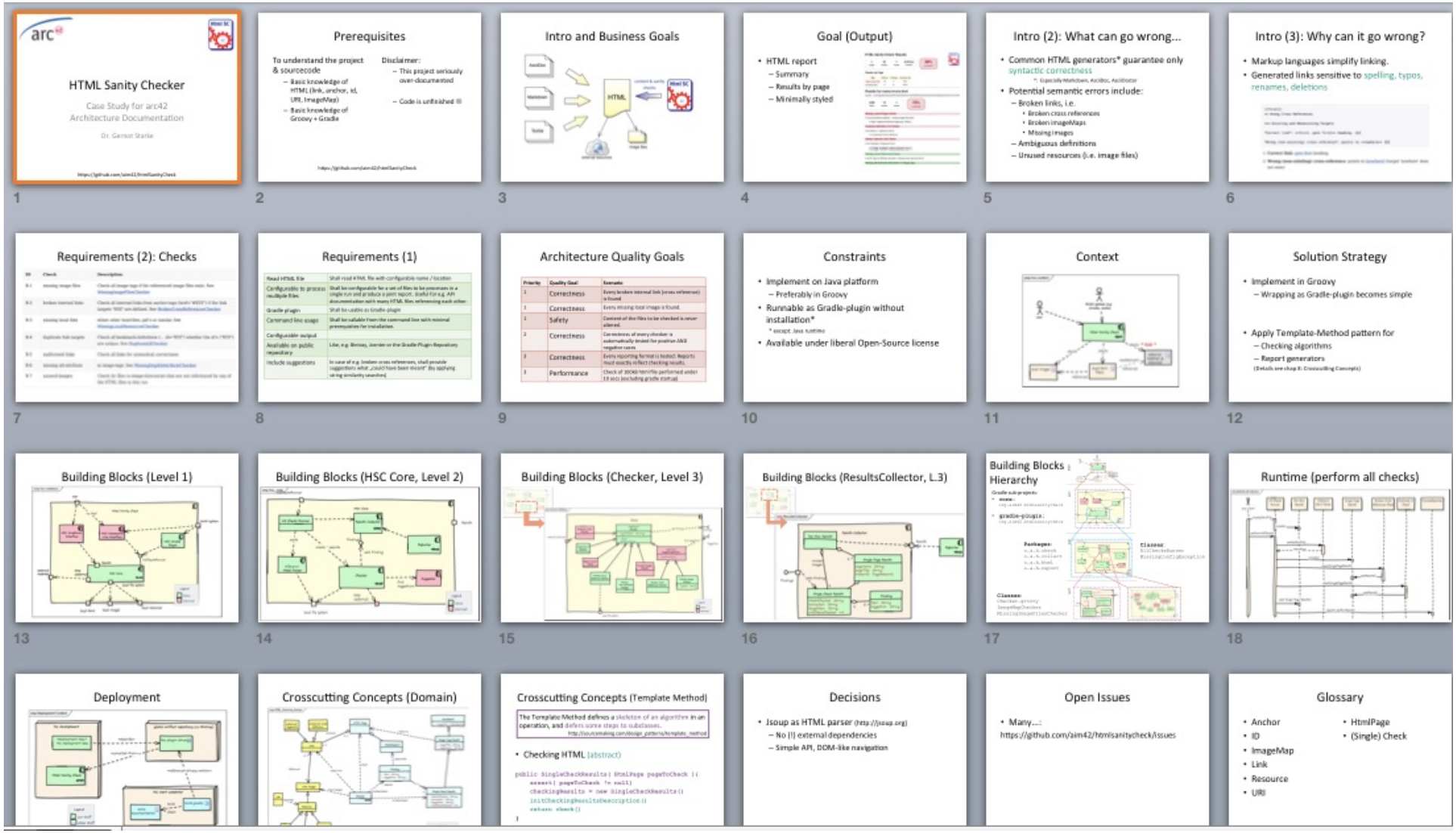
Beispiel (Datenmigration, 4000+ PT)



Beispiel (CRM-System, 2000+ PT)

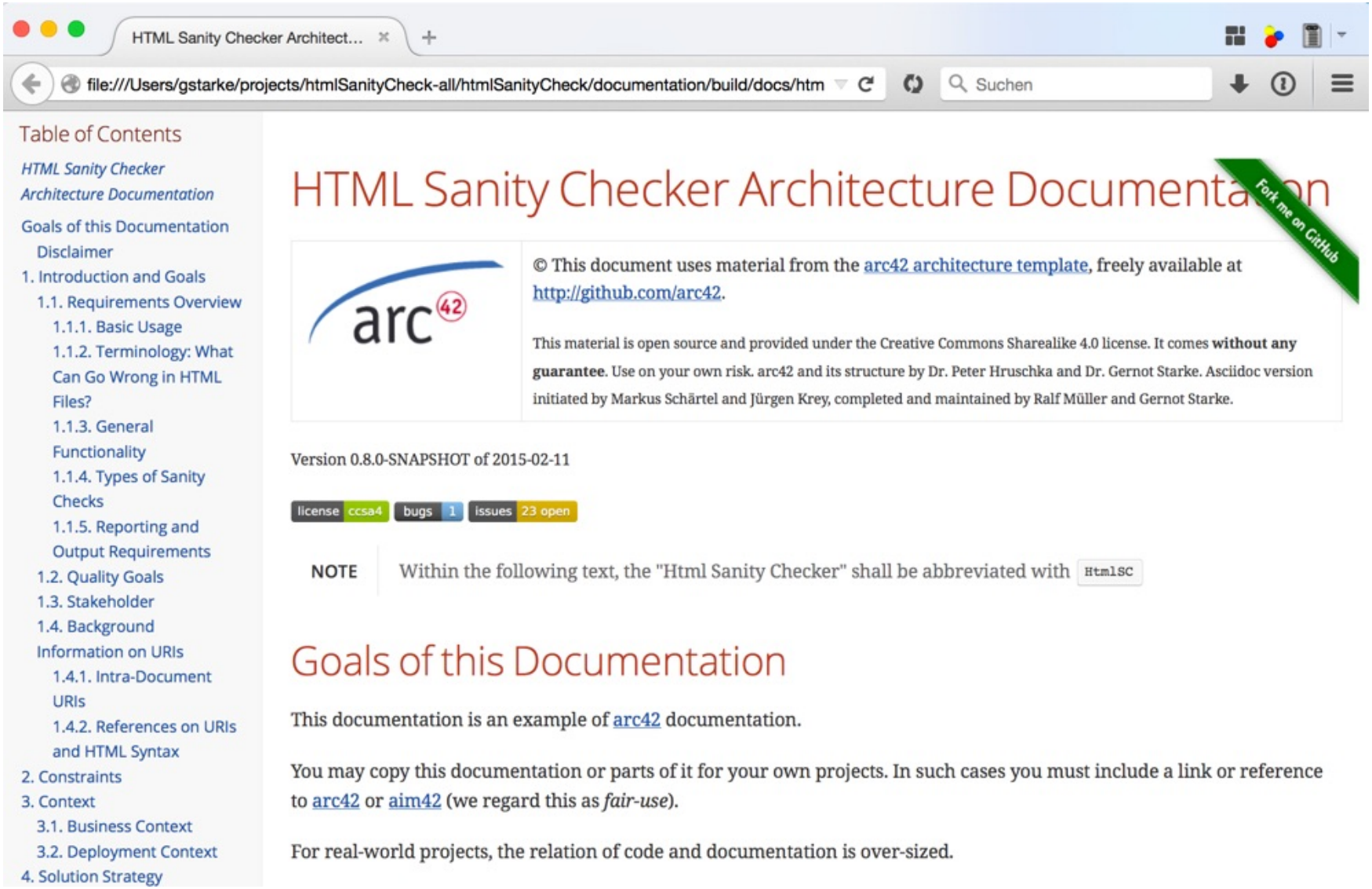


Beispiel „htmlSanityCheck“



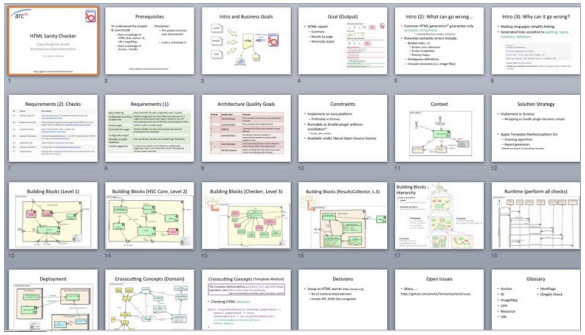
https://github.com/aim42/htmlsanitycheck

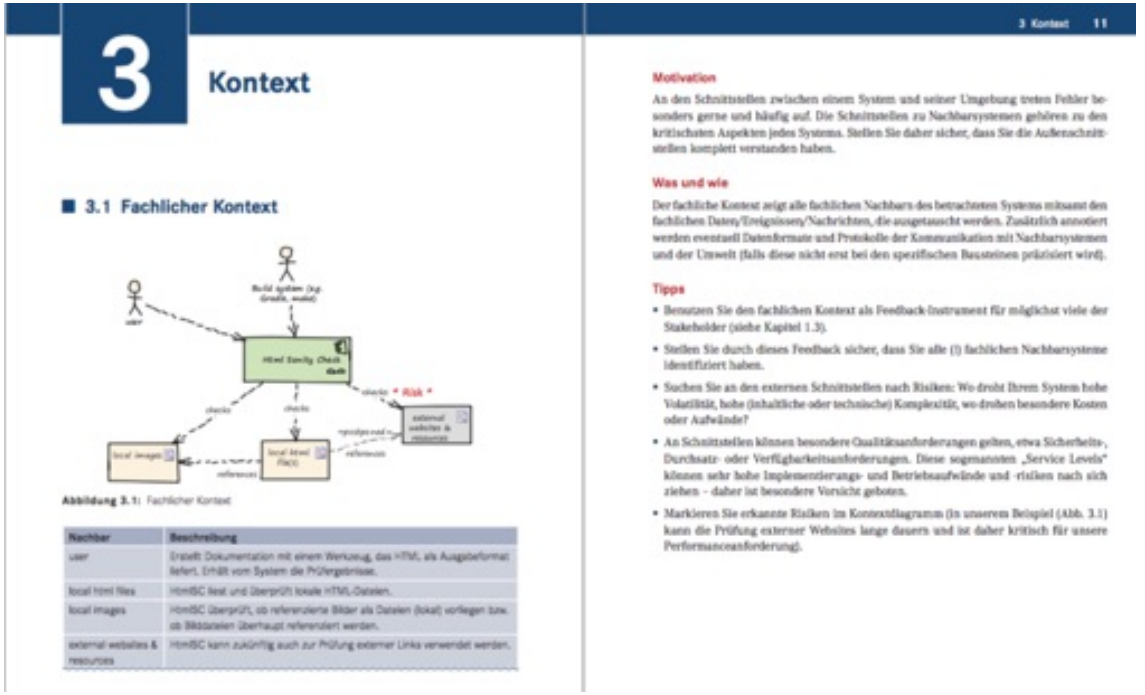
Beispiel „htmlSanityCheck“ (2)



Mit arc42 starten?

Mit arc42 starten (Tag 1)





3 Kontext

3 Kontext

3.1 Fachlicher Kontext

Abbildung 3.1: Fachlicher Kontext

Nachbar	Beschreibung
user	Erstellt Dokumentation mit einem Werkzeug, das HTML als Ausgabeformat liefert. Erhält vom System die Prüfergebnisse.
local html files	HtmlSC liest und überprüft lokale HTML-Dateien.
local images	HtmlSC überprüft, ob referenzierte Bilder als Dateien (lokal) vorliegen bzw. ob Bilddateien überhaupt referenziert werden.
external websites & resources	HtmlSC kann zukünftig auch zur Prüfung externer Links verwendet werden.

Motivation

An den Schnittstellen zwischen einem System und seiner Umgebung treten Fehler besonders gerne und häufig auf. Die Schnittstellen zu Nachbarsystemen gehören zu den kritischsten Aspekten jedes Systems. Stellen Sie daher sicher, dass Sie die Außenschnittstellen komplett verstanden haben.

Was und wie

Der fachliche Kontext zeigt alle fachlichen Nachbarn des betrachteten Systems mitsamt den fachlichen Daten/Ereignissen/Nachrichten, die ausgetauscht werden. Zusätzlich annotiert werden eventuell Datenformate und Protokolle der Kommunikation mit Nachbarsystemen und der Umwelt (falls diese nicht erst bei den spezifischen Bausteinen präzisiert wird).

Tipps

- Benutzen Sie den fachlichen Kontext als Feedback-Instrument für möglichst viele der Stakeholder (siehe Kapitel 1.3).
- Stellen Sie durch dieses Feedback sicher, dass Sie alle (!) fachlichen Nachbarsysteme identifiziert haben.
- Suchen Sie an den externen Schnittstellen nach Risiken: Wo droht Ihrem System hohe Volatilität, hohe (inhaltliche oder technische) Komplexität, wo drohen besondere Kosten oder Aufwände?
- An Schnittstellen können besondere Qualitätsanforderungen gelten, etwa Sicherheits-, Durchsatz- oder Verfügbarkeitsanforderungen. Diese sogenannten „Service Levels" können sehr hohe Implementierungs- und Betriebsaufwände und -risiken nach sich ziehen – daher ist besondere Vorsicht geboten.
- Markieren Sie erkannte Risiken im Kontextdiagramm (in unserem Beispiel (Abb. 3.1) kann die Prüfung externer Websites lange dauern und ist daher kritisch für unsere Performanceanforderung).

Beispiele!

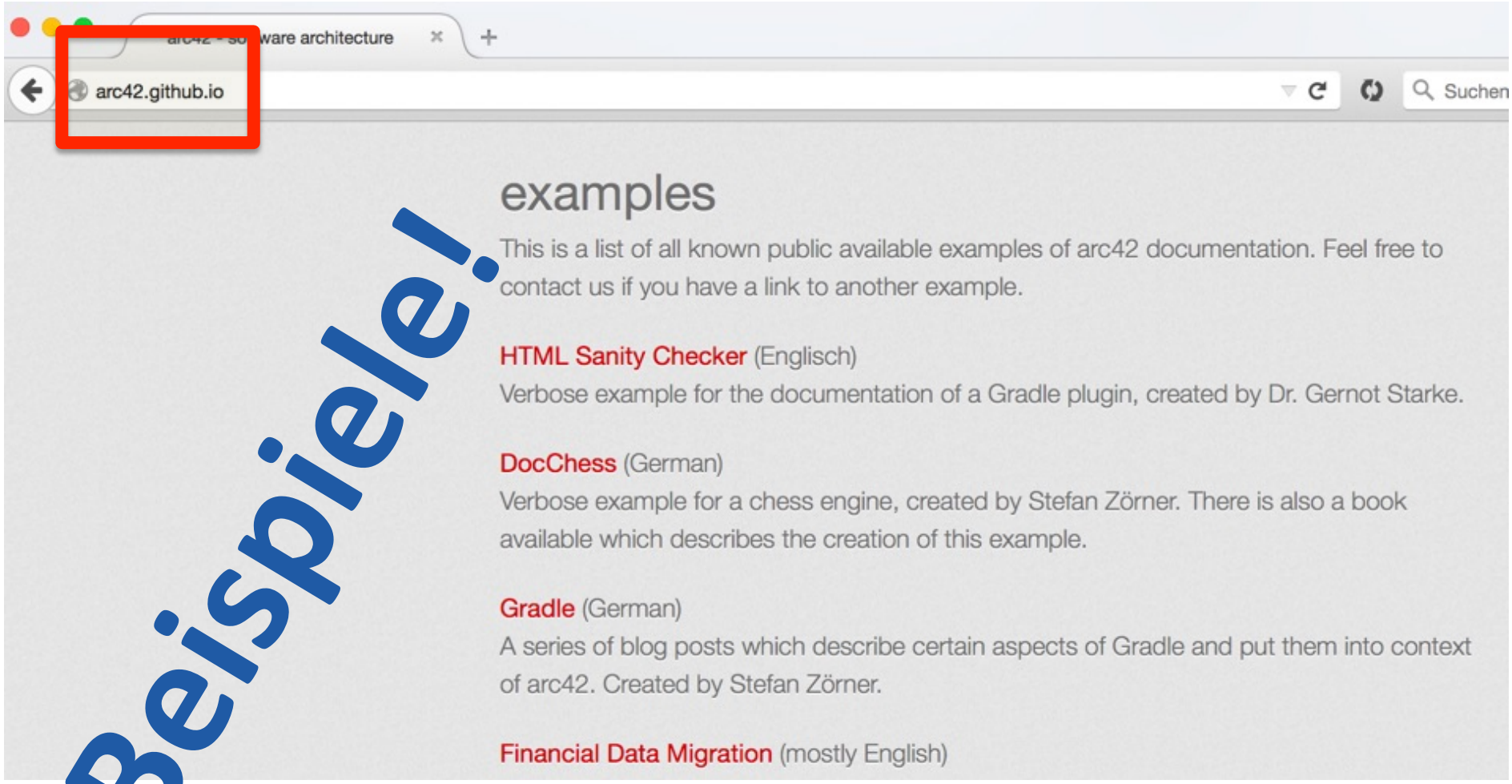
http://aim42.github.io/htmlSanityCheck/

arc42

innoQ-Stand *

* gerne im Tausch gegen vcard

Mit arc42 starten (Tag 1)



Beispiele!

Mit arc42 starten (Tag 1)

Willkommen **DokChess** Das Buch Selber starten? Informationen Unterstützung? Kontakt

arc42 Quelltexte Test-Suite javadoc

- > 1. Einführung
- > 2. Randbedingungen
- > 3. Kontextabgrenzung
- > 4. Lösungsstrategie
- > 5. Bausteinsicht
- > 6. Laufzeitsicht
- 7. Verteilungssicht
- 8. Konzepte
- 9. Entscheidungen
- 10. Szenarien
- 11. Risiken
- 12. Glossar

DokChess als Beispiel für arc42

Diese Seiten beschreiben die Architektur des Schach-Programmes DokChess. Ich habe es als Anschauungsmaterial für Vorträge und Seminare rund um Softwarearchitektur und -entwurf konzipiert und implementiert. Es dient als Fallbeispiel in meinem Buch über Architekturdokumentation.

Architekturüberblick DokChess

Dieser Architekturüberblick lässt Sie die maßgeblichen Entwurfsentscheidungen nachvollziehen. Er zeigt die Struktur der Lösung und das Zusammenspiel zentraler Elemente. Die Gliederung der Inhalte erfolgt nach der arc42-Vorlage. Zielgruppe dieses Überblicks sind in erster Linie Softwarearchitekten, die Anregungen und Beispiele suchen, wie man Architekturentwürfe angemessen dokumentieren kann. Entwickler, die selbst ein Schachprogramm schreiben wollen, erhalten wertvolle Tipps und lernen en passant einiges über methodische Softwarearchitektur.

Idee + Ausführung: Stefan Zörner (http://embarc.de)

Mit arc42 starten (Tag 1)

Mehr Beispiele!

http://confluence.arc42.org:

- CRM-System (2000+ PT)
- Datenmigration (4000+ PT)

Mit arc42 starten (Tag 1, 13:30h)...

neue Systeme

1. Kontext
2. Qualitätsziele
3. Lösungsstrategie
4. Konzepte
    1. Domäne
    2. Persistenz, UI etc.

bestehende Systeme

1. Bausteine (Code-Struktur) Level 1
2. Konzepte + typische Lösungsmuster
3. Externe Schnittstellen (= Kontext)

Mit arc42 im Projekt?

Mit arc42 im Projekt...

Team

Projektdokumentation

„Gärtner“

Systemdokumentation

Mit arc42 im Projekt...

Team

Projektdokumentation

~arc42 „locker“

Diskussionen

Implementation-Guide,

Tasks / Issues

„Gärtner“

Systemdokumentation

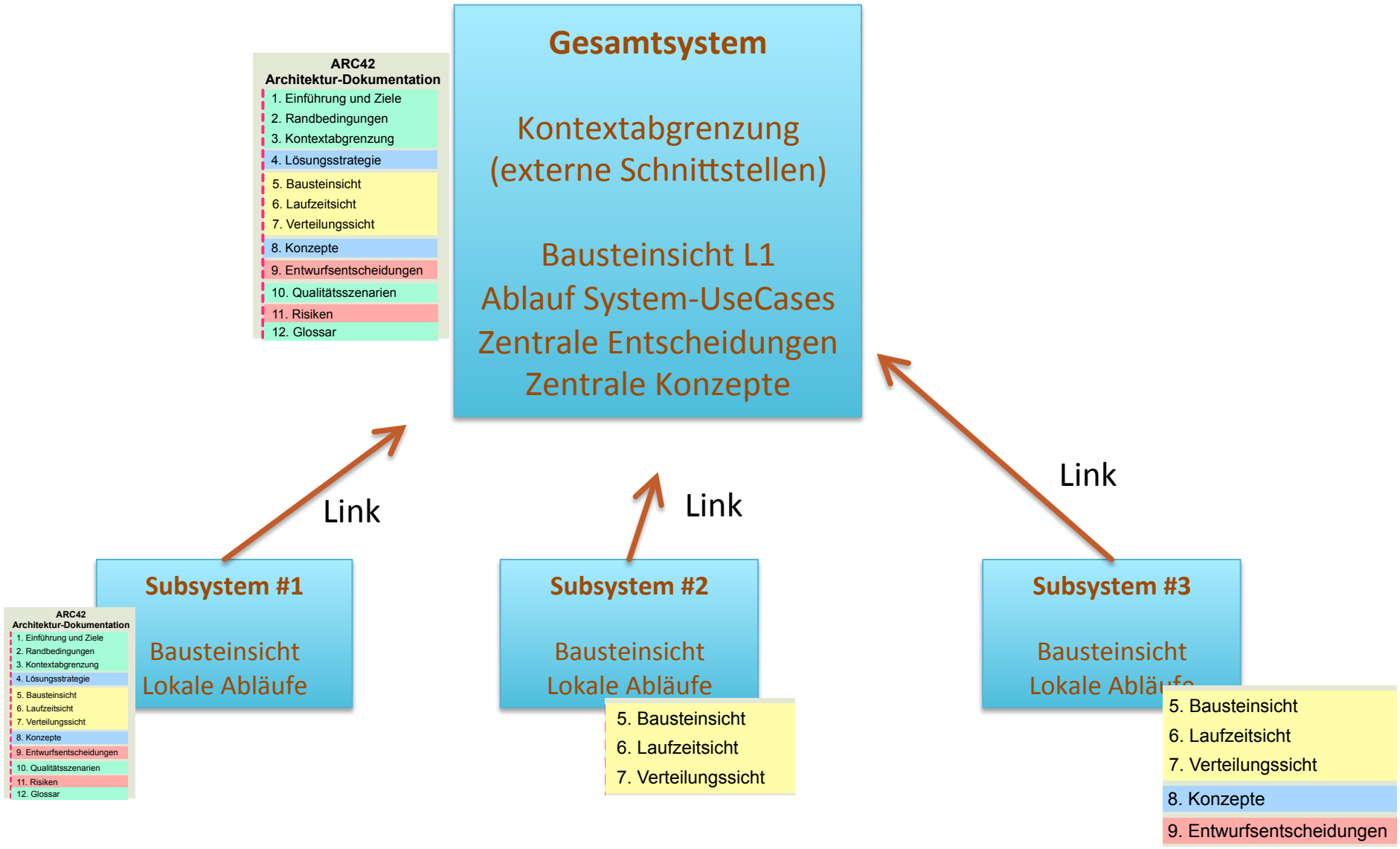
1. Einführung und Ziele
2. Randbedingungen
3. Kontextabgrenzung
4. Lösungsstrategie
5. Bausteinsicht
6. Laufzeitsicht
7. Verteilungssicht
8. Konzepte
9. Entwurfsentscheidungen
10. Qualitätsszenarien
11. Risiken
12. Glossar

Weitere relevante Infos...

(Betrieb/Admin, Test, Release...)

Große Systeme mit arc42?

arc42 und größere Systeme



Kleine Systeme mit arc42?

Kleine Systeme...

Leeres Template == 26 Seiten

Übertrieben für kleine Projekte, oder?

Kleine Systeme...

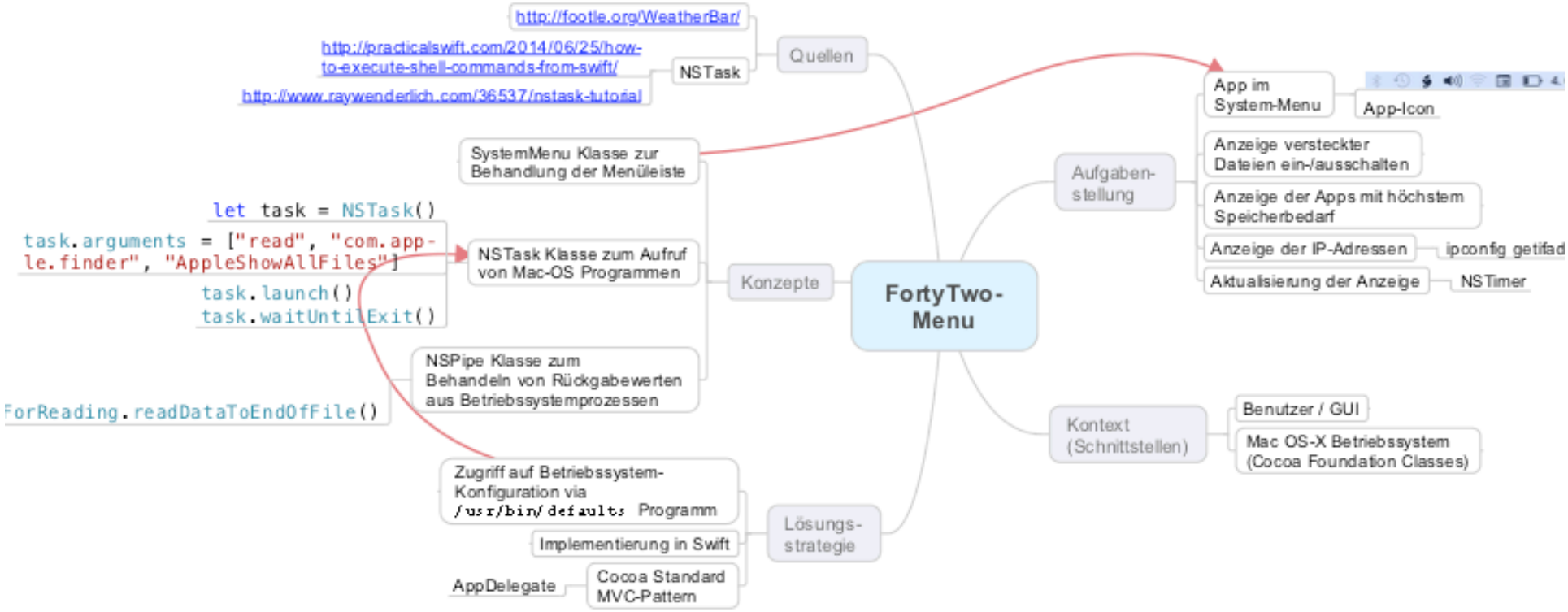
Leeres Template == 26 Seiten
Übertrieben für kleine Projekte, oder?

arc42 == **Schrank**, nicht Formular

SUPER für kleine Projekte:

- *timeboxed* verwenden
- essentielle Dinge einbringen

Beispiel „kleines System“

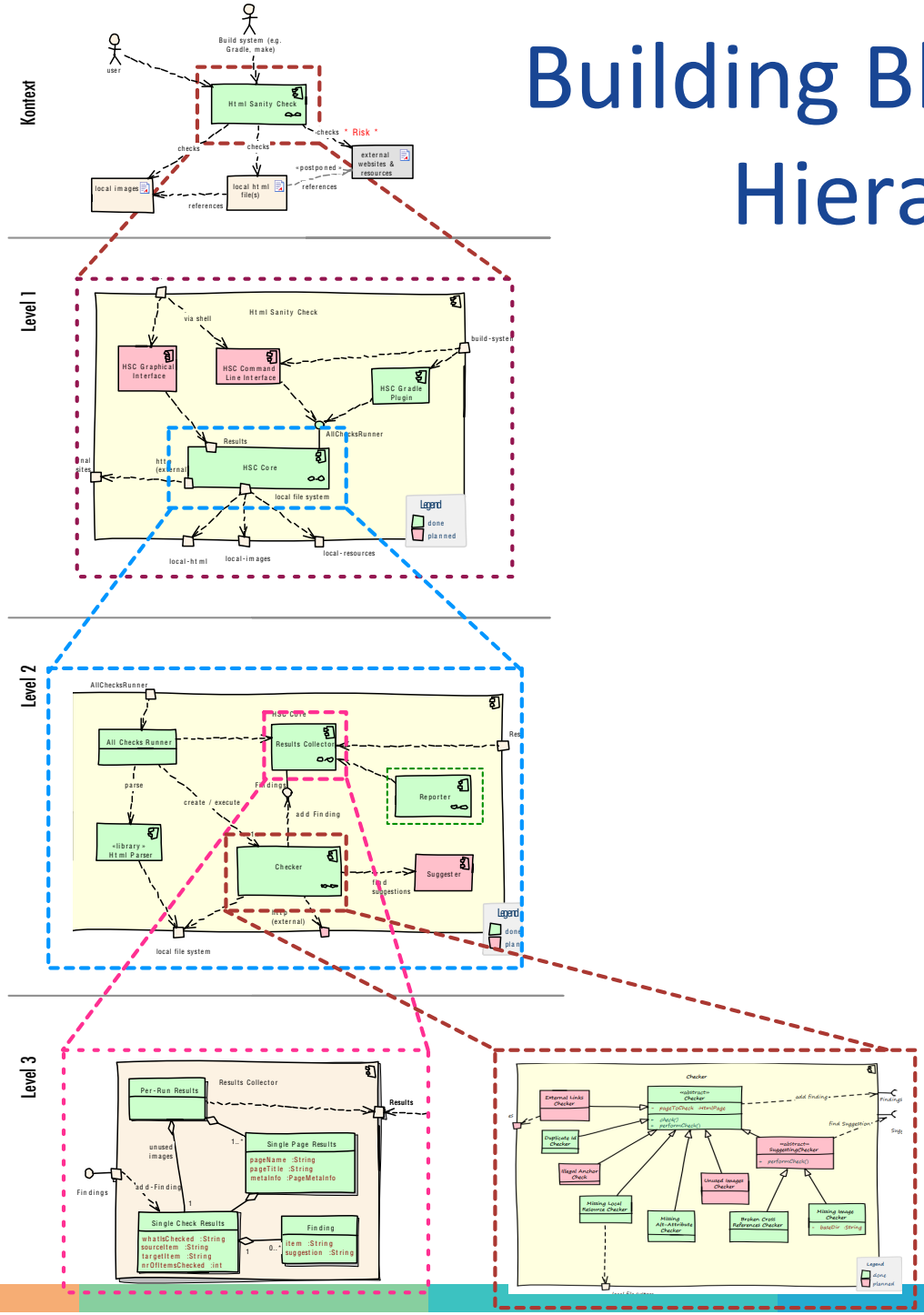


Code und Bausteinsicht?

Building Blocks Hierarchy

Gradle sub-projects:

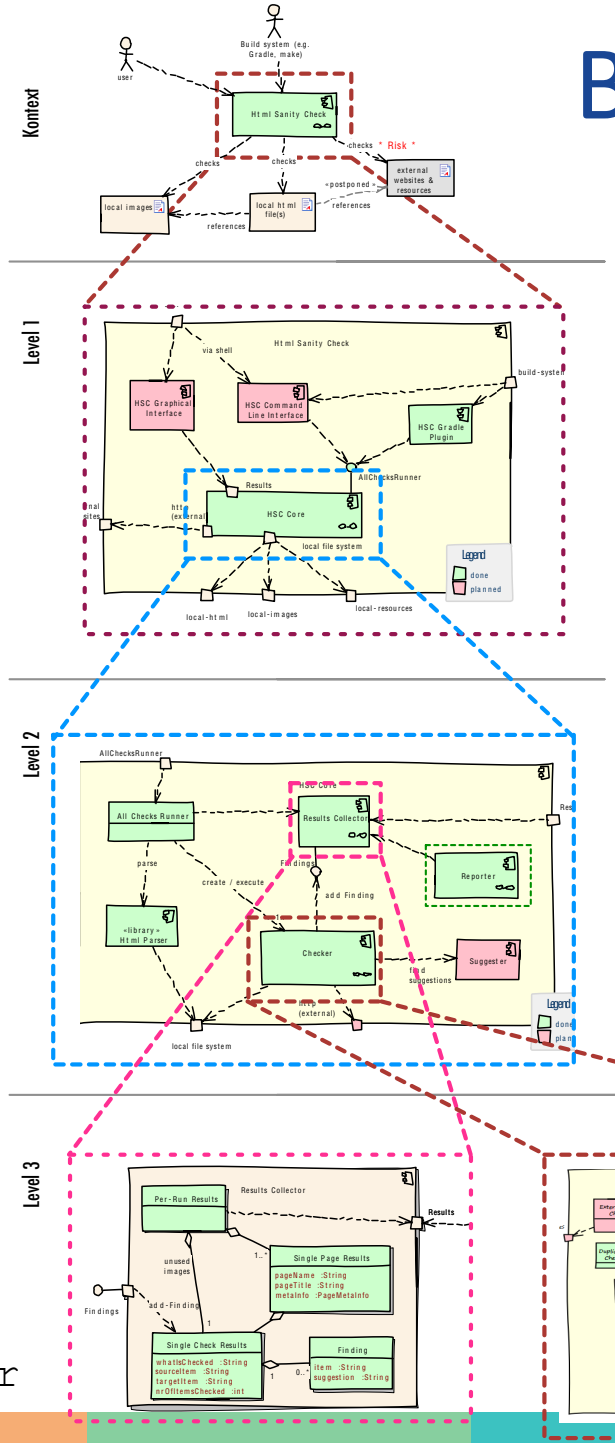
- **`core:`**
  `org.aim42.htmlsanitycheck`
- **`gradle-plugin:`**
  `org.aim42.htmlsanitycheck`



Building Blocks Hierarchy

Gradle sub-projects:

- **`core:`**
  `org.aim42.htmlsanitycheck`
- **`gradle-plugin:`**
  `org.aim42.htmlsanitycheck`

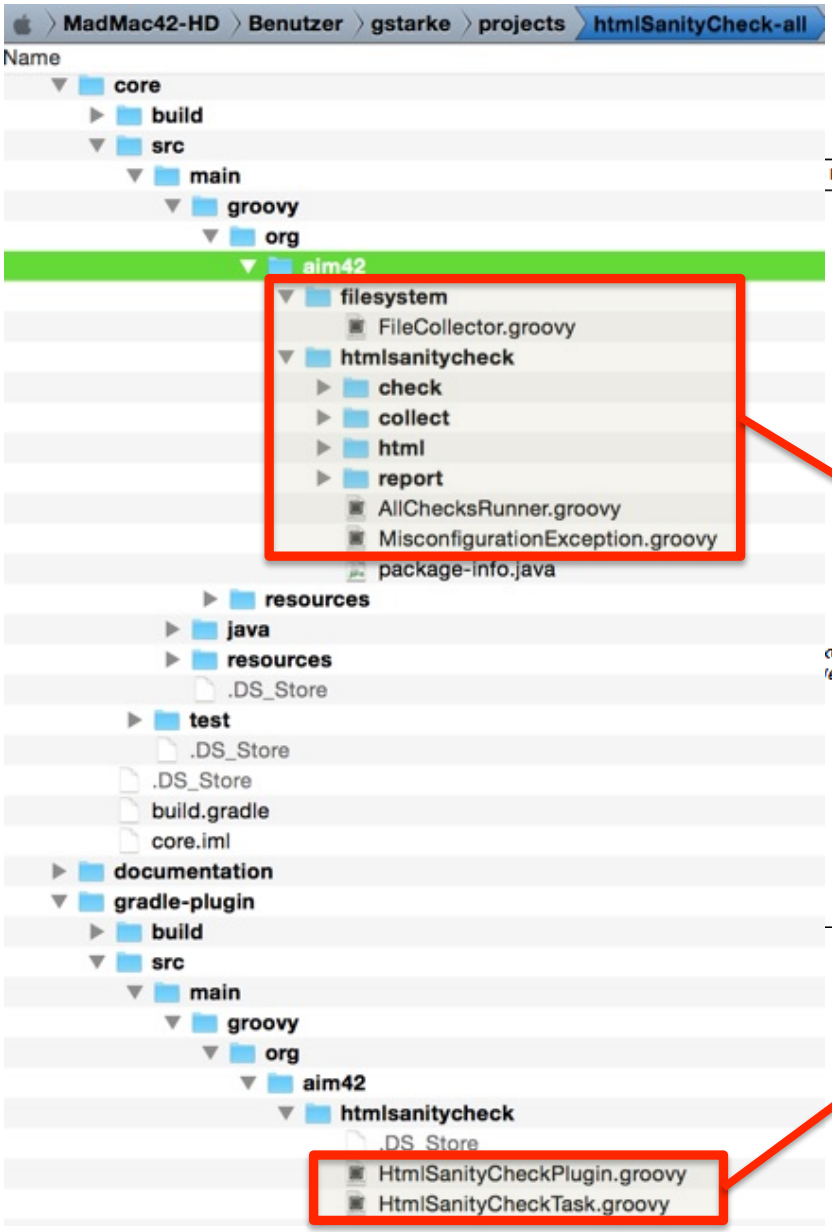


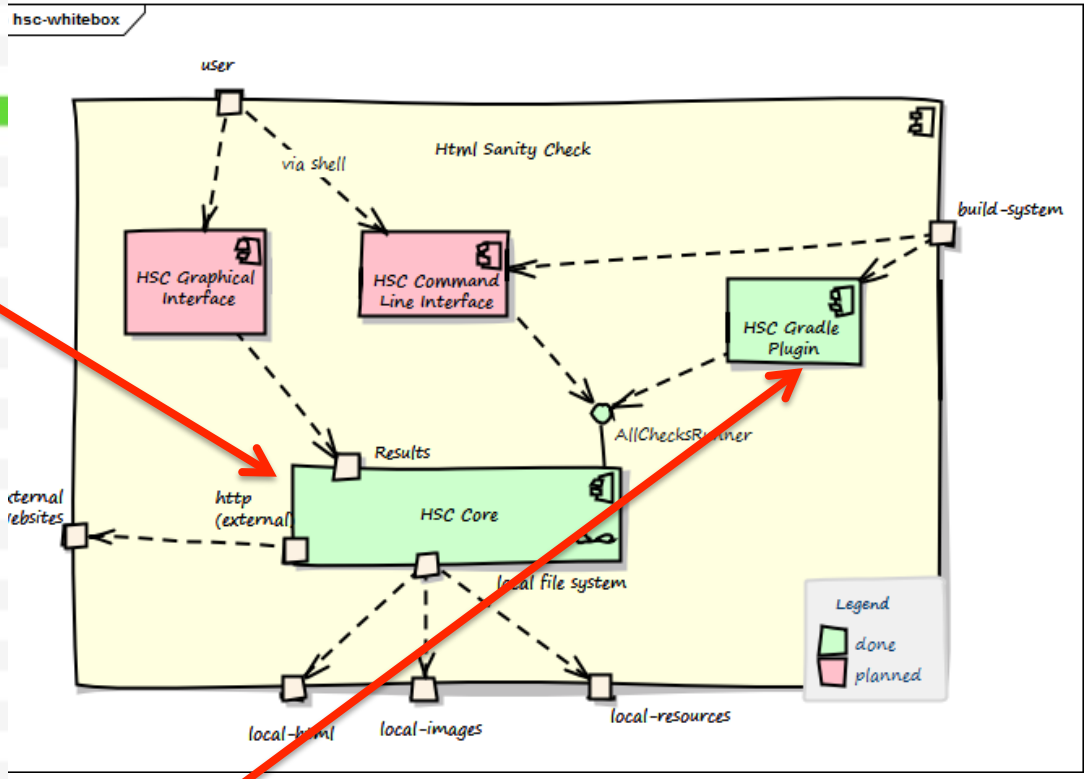
`Packages`:
`o.a.h.check`
`o.a.h.collect`
`o.a.h.html`
`o.a.h.report`

`Classes`:
`AllChecksRunner`
`MissingConfigException`

`Classes`:
`Checker.groovy`
`ImageMapChecker`
`MissingImageFilesChecker`

Level-1: Codestruktur im Großen

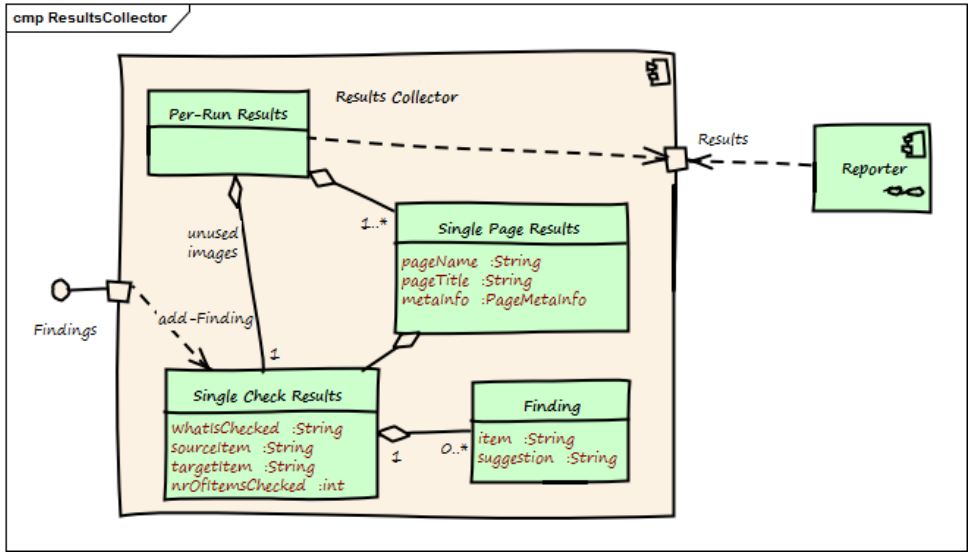




arc42

Code in der Dokumentation?

Code in Doku: Beispiel...



Wichtig: **NCP**

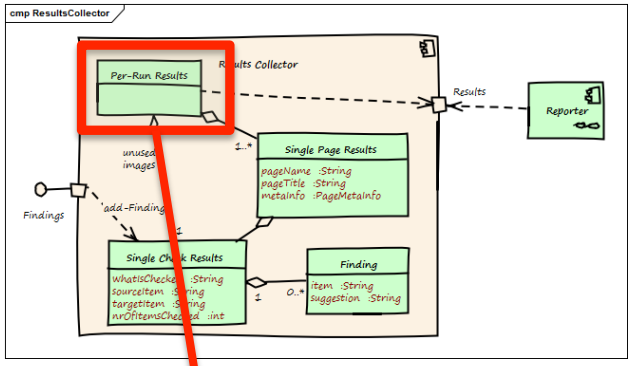
(never copy-paste)

Contained Blackboxes

Table 13. ResultsCollector building blocks

Per-Run Results	results for potentially many Html pages/documents.
Single-Page-Results	results for a single page
Single-Check-Results	results for a single type of check (e.g. missing-images check)
Finding	a single finding, (e.g. "image 'logo.png' missing"). Can hold suggestions and (planned for future releases) the responsible html element.

Code in Doku: Beispiel...



Interface RunResults

```
public interface RunResults {

    // returns results for all pages which have been checked
    public ArrayList<SinglePageResults> getResultsForAllPages()

    // how many pages were checked in this run?
    public int nrOfPagesChecked()

    // how many checks were performed in all?
    public int nrOfChecksPerformedOnAllPages()

    // how many findings (errors and issues) were found in all?
    public int nrOfFindingsOnAllPages()

    // how long took checking (in milliseconds)?
    public Long checkingTookHowManyMillis()
}
```

asciidoc-Source

```
[source, groovy]
.Interface RunResults
----
include::{coresourcepath}/htmlsanitycheck/collect/
RunResults.groovy[tags=RunResultInterface]
----
```

Konzepte!

Was ist ein Konzept?

Beispiele im Bauwesen:

- Fenster und –griffe
- Heizung / Klima
- Flach-/Walm-/Spitzdach

Was ist ein Konzept?

Plan zur Lösung von

- mehrfach auftretenden,
- übergreifenden

Problemen.

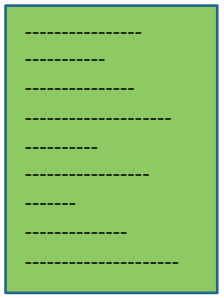
Beispiele:

- Persistenz
- Reporting
- Batch
- Validierung
- ....

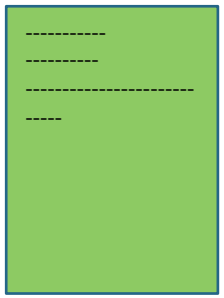
Themen finden

Gliederung nach S. Zörner

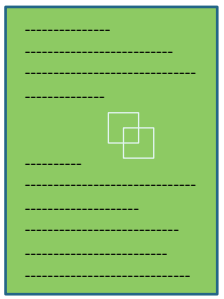
- Architekturmuster
- Entwicklung / Weiterentwicklung
  - Codegenerierung, Buildmanagment, Tests, Migration, Konfiguration, Modellierung
- Unter-der-Haube
  - Persistenz, Verteilung, Sicherheit, Transaktionen, Sessions, Caching, Parallelisierung, Threading, Geschäftsregeln, Batchverarbeitung
- Interaktion
  - UI, i18n, Ergonomie, Validierung, Barrierefreiheit, Plausibilisierung, Ausnahme-/Fehlerbehandlung, Ablaufsteuerung, Kommunikation, Integration, Reporting
- Betrieb



Identifizieren

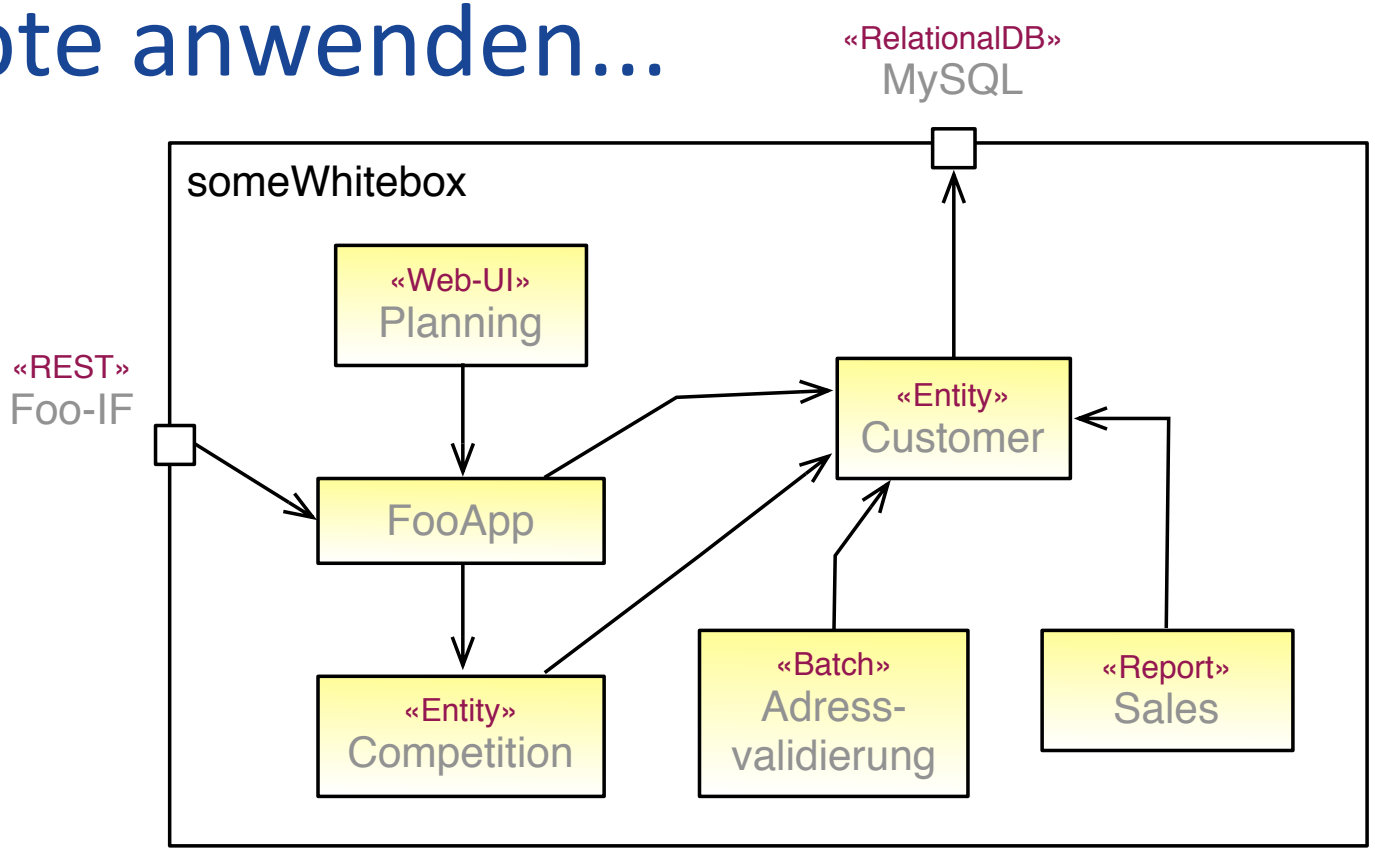


Priorisieren



ausführen

Konzepte anwenden...



Bausteinsicht:

- Zeigt Stereotypen
  (=> Verweise auf Konzepte)

Konzepte:

- erklären Implementierung, mit Beispielen

Werkzeuge!

Anforderungen an Tools

- Text + Tabellen
- Diagramme
- Verweise

- Multi-User
- Änderungshistorie
- Versioniert

Variante 1: plain-Wiki

- Text + Tabellen
- Diagramme
- Verweise

- Multi-User
- Änderungshistorie
- Versioniert

Variante 1b: plain-Wiki++

- Text + Tabellen
- Diagramme
- Verweise

Zeichen-Tool (Visio, Graffle & Co)

- Multi-User
- Änderungshistorie
- Versioniert

Variante 1c: Wiki + Grafik-Plugin

- Text + Tabellen
- Diagramme
- Verweise

Etwa: Gliffy
(für Confluence)

- Multi-User
- Änderungshistorie
- Versioniert

Variante 2: Wiki + UML-Tool

- Text + Tabellen
- Diagramme
- Verweise

UML-Tool
(Enterprise-Architect,
MagicDraw,
Visual-Paradigm & Co)

- Multi-User
- Änderungshistorie
- Versioniert

- Usability
- Akzeptanz

Variante 2: Plaintext + UML-Tool

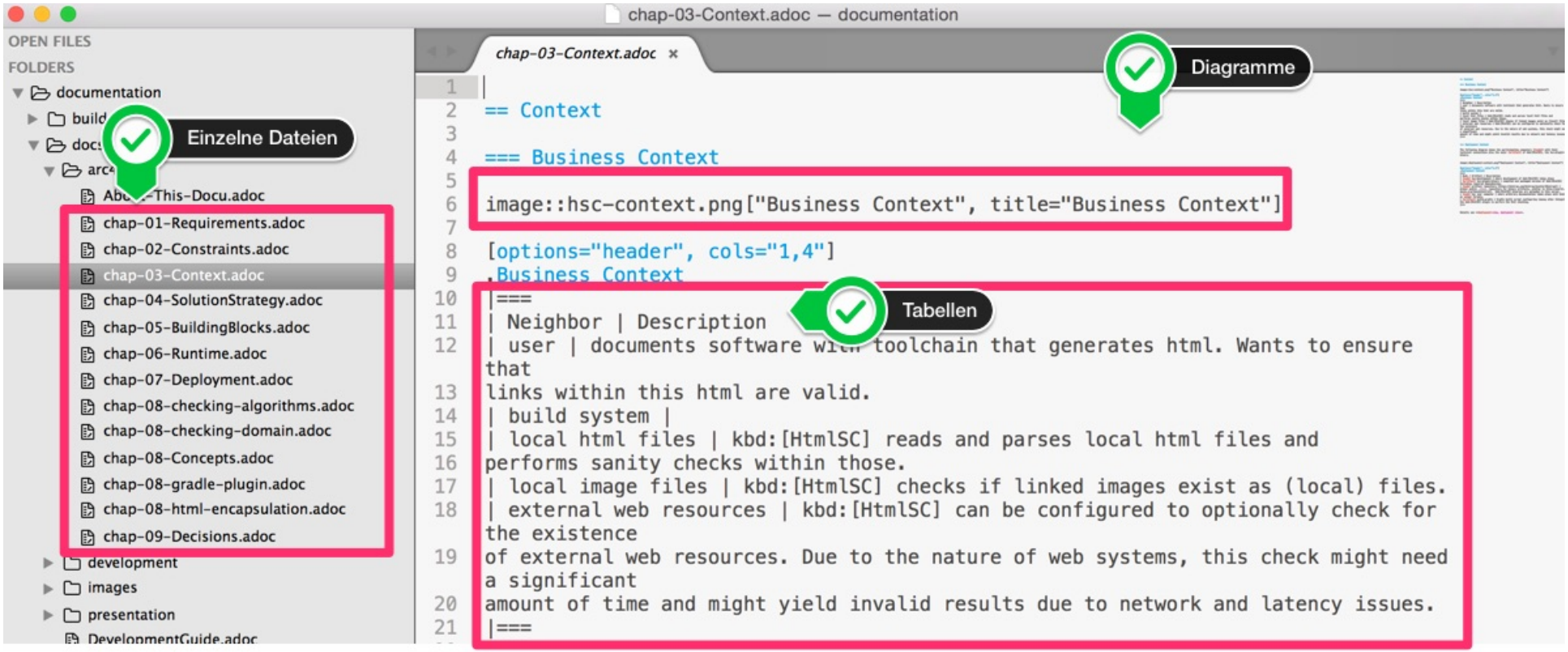
- Geek-Faktor

AsciiDoc, Markdown

- Text + Tabellen
- Diagramme
- Verweise

- Multi-User
- Änderungshistorie
- Versioniert

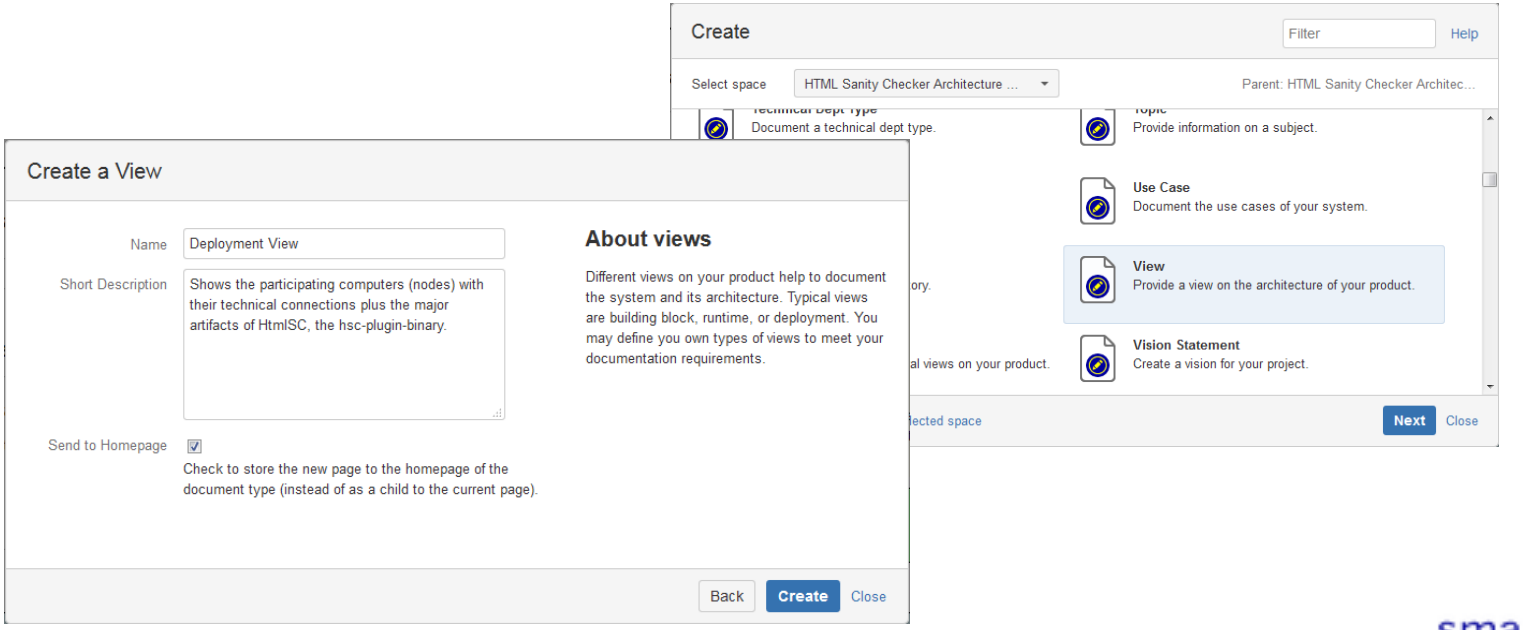
AsciiDoc...



- Plaintext (git, diff)
- Build z.B. mit Gradle
- Zielformate: html, epub, pdf

Confluence + ProjectDoc (kommerziell)

- Idee: flüssigeres Arbeiten dank Wizards

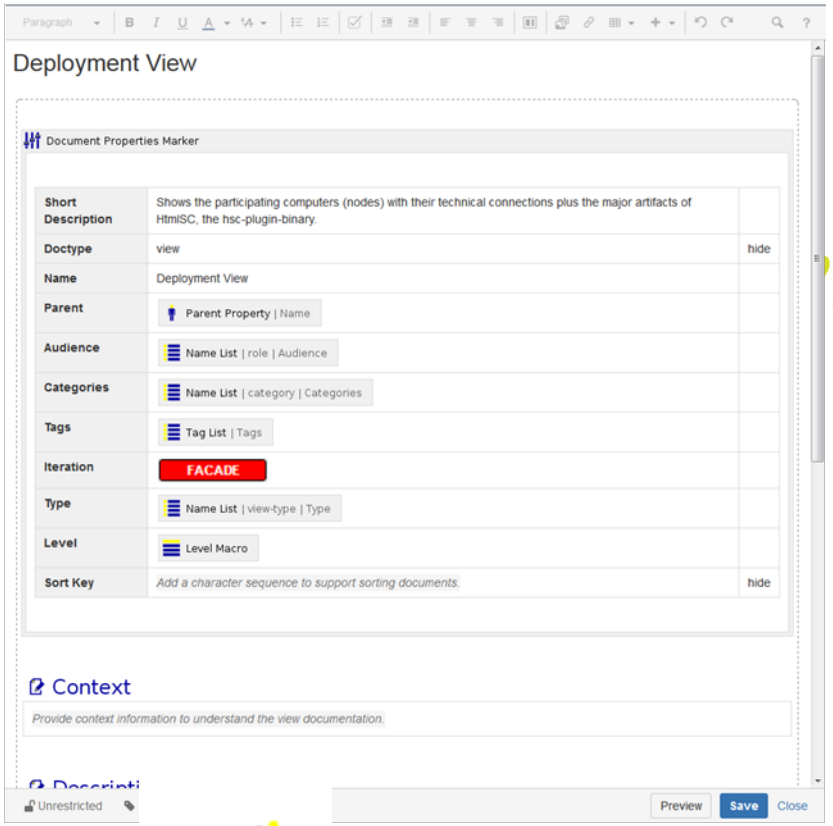


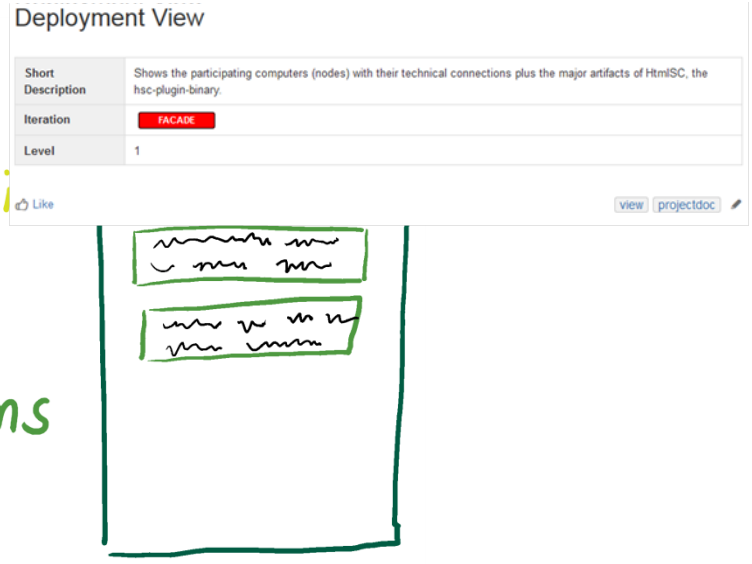
smartics

https://www.smartics.de/go/arc42

arc42 mit projectdoc-Blueprints

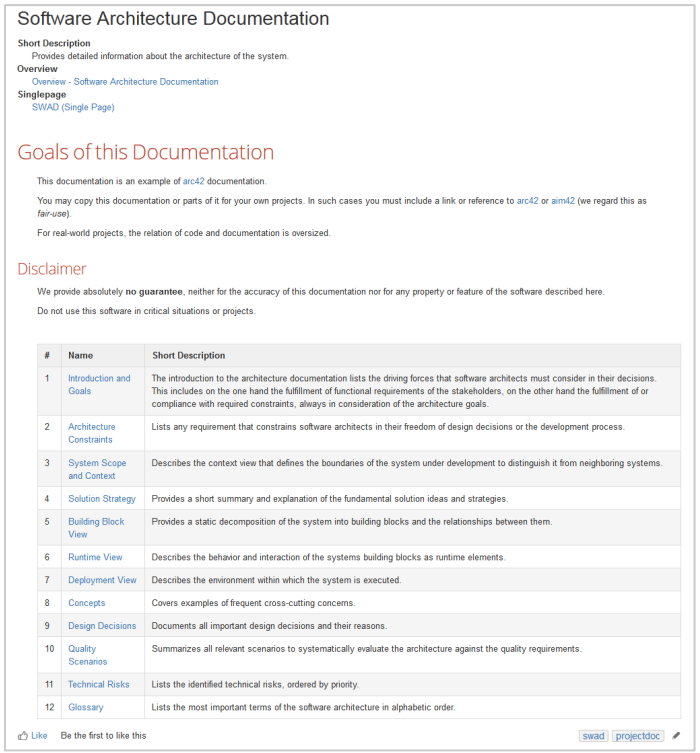
- Vorlagen als Schreibhilfen





smartics

HTML Sanity Checker mit projectdoc



Software Architecture Documentation

Short Description
Provides detailed information about the architecture of the system.
Overview
Overview - Software Architecture Documentation
Singlepage
SWAD (Single Page)

Goals of this Documentation

This documentation is an example of arc42 documentation.

You may copy this documentation or parts of it for your own projects. In such cases you must include a link or reference to arc42 or aim42 (we regard this as *fair-use*).

For real-world projects, the relation of code and documentation is oversized.

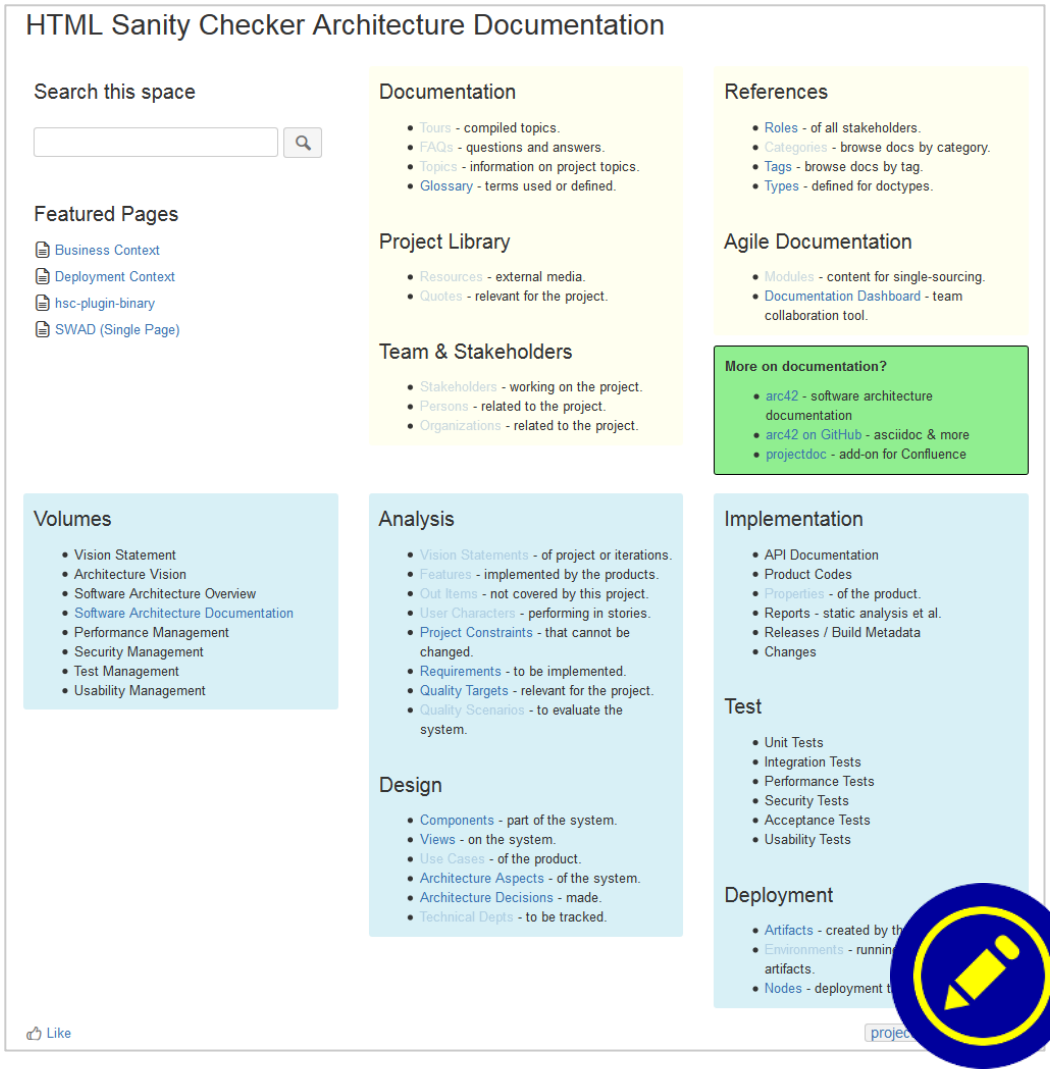
Disclaimer

We provide absolutely **no guarantee**, neither for the accuracy of this documentation nor for any property or feature of the software described here.

Do not use this software in critical situations or projects.

#	Name	Short Description
1	Introduction and Goals	The introduction to the architecture documentation lists the driving forces that software architects must consider in their decisions. This includes on the one hand the fulfillment of functional requirements of the stakeholders, on the other hand the fulfillment of or compliance with required constraints, always in consideration of the architecture goals.
2	Architecture Constraints	Lists any requirement that constrains software architects in their freedom of design decisions or the development process.
3	System Scope and Context	Describes the context view that defines the boundaries of the system under development to distinguish it from neighboring systems.
4	Solution Strategy	Provides a short summary and explanation of the fundamental solution ideas and strategies.
5	Building Block View	Provides a static decomposition of the system into building blocks and the relationships between them.
6	Runtime View	Describes the behavior and interaction of the systems building blocks as runtime elements.
7	Deployment View	Describes the environment within which the system is executed.
8	Concepts	Covers examples of frequent cross-cutting concerns.
9	Design Decisions	Documents all important design decisions and their reasons.
10	Quality Scenarios	Summarizes all relevant scenarios to systematically evaluate the architecture against the quality requirements.
11	Technical Risks	Lists the identified technical risks, ordered by priority.
12	Glossary	Lists the most important terms of the software architecture in alphabetic order.

Like Be the first to like this swad projectdoc



HTML Sanity Checker Architecture Documentation

Search this space

Featured Pages
- Business Context
- Deployment Context
- hsc-plugin-binary
- SWAD (Single Page)

Documentation
- Tours - compiled topics.
- FAQs - questions and answers.
- Topics - information on project topics.
- Glossary - terms used or defined.

Project Library
- Resources - external media.
- Quotes - relevant for the project.

Team & Stakeholders
- Stakeholders - working on the project.
- Persons - related to the project.
- Organizations - related to the project.

References
- Roles - of all stakeholders.
- Categories - browse docs by category.
- Tags - browse docs by tag.
- Types - defined for doctypes.

Agile Documentation
- Modules - content for single-sourcing.
- Documentation Dashboard - team collaboration tool.

More on documentation?
- arc42 - software architecture documentation
- arc42 on GitHub - asciidoc & more
- projectdoc - add-on for Confluence

Volumes
- Vision Statement
- Architecture Vision
- Software Architecture Overview
- Software Architecture Documentation
- Performance Management
- Security Management
- Test Management
- Usability Management

Analysis
- Vision Statements - of project or iterations.
- Features - implemented by the products.
- Out Items - not covered by this project.
- User Characters - performing in stories.
- Project Constraints - that cannot be changed.
- Requirements - to be implemented.
- Quality Targets - relevant for the project.
- Quality Scenarios - to evaluate the system.

Design
- Components - part of the system.
- Views - on the system.
- Use Cases - of the product.
- Architecture Aspects - of the system.
- Architecture Decisions - made.
- Technical Depts - to be tracked.

Implementation
- API Documentation
- Product Codes
- Properties - of the product.
- Reports - static analysis et al.
- Releases / Build Metadata
- Changes

Test
- Unit Tests
- Integration Tests
- Performance Tests
- Security Tests
- Acceptance Tests
- Usability Tests

Deployment
- Artifacts - created by th
- Environments - runnin artifacts.
- Nodes - deployment t

Like

Variante 3: UML-Tool

- Text + Tabellen
- **Diagramme**
- Verweise

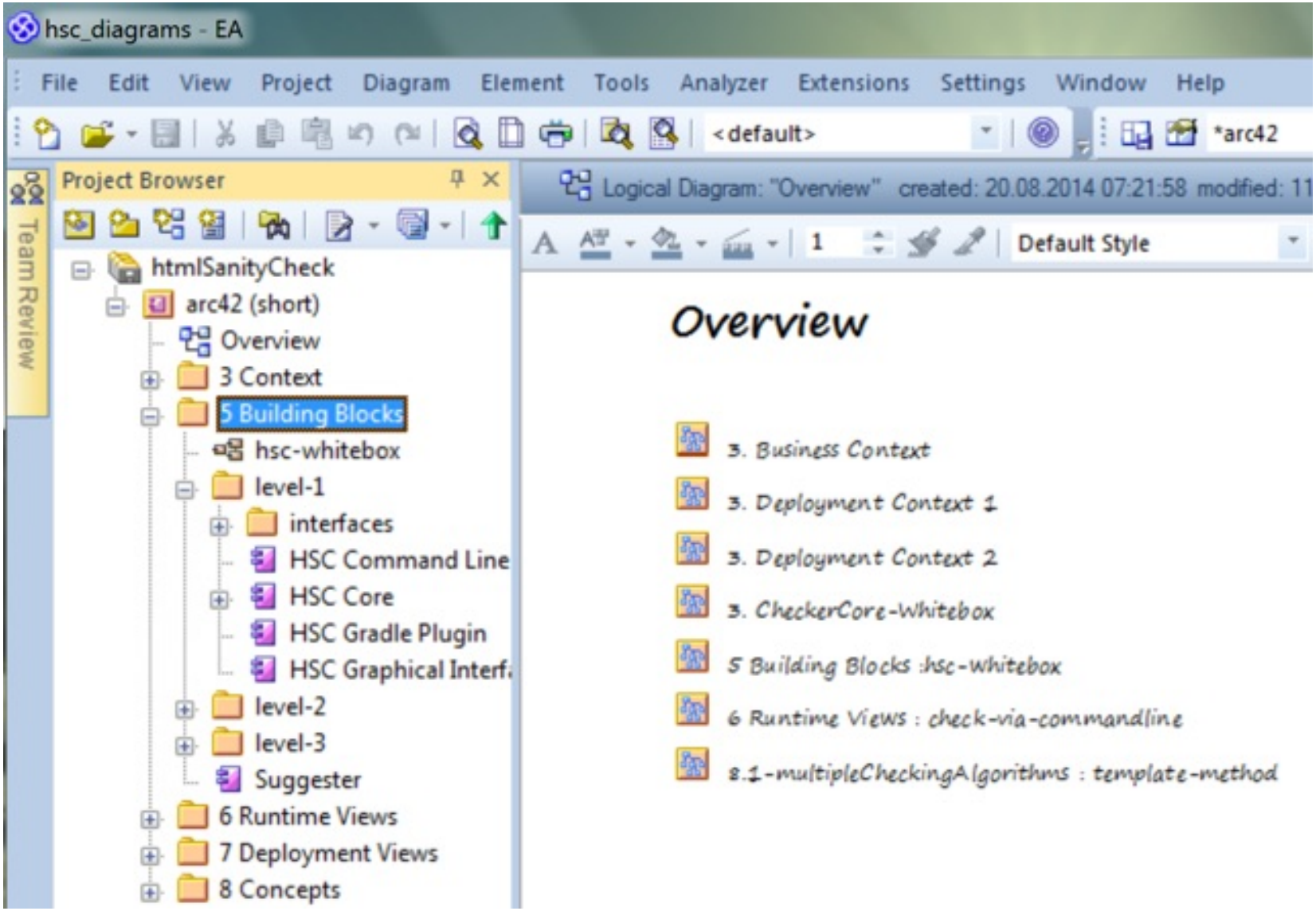
Modelle

- Geek-Faktor

Enterprise-Architect,
Visual Paradigm etc.

- Multi-User
- Änderungshistorie
- Versioniert

Beispiel: arc42 mit EA



UML-Tool + Export-Automatisierung

- Export aus Enterprise-Architect™
  - Skript in github.com/arc42 vorhanden

- Export aus VisualParadigm™
  - Export möglich
    - bislang kein OpenSource Skript verfügbar

Wohin damit?

Wohin mit Domain-Model?

1. (gut) Querschnittliche Konzepte

2. (ok) Bausteinsicht („Domain-Component“)

3. (hhm) Eigenes (neues) Kapitel

Wohin mit UI/Layout, Masken oder Screenflow?

1. Querschnittliche Konzepte

2. Eigene Sicht
   - (== neues Kapitel)

Kata mit arc42

(„Hands-on Workshop“)

Ziele einer Architektur-Kata

- Gemeinsames Verständnis
- Aufgaben „erproben“

- Aspekte von „Architektur“ erarbeiten

Kata mit arc42

Aufgabe

Anforderungen

Qualitätsziele

Kontext

Lösungsidee

Bausteine

Konzepte

Subprojekte!

Qualitätsanforderungen

wesentliche
Architekturaufgaben

Beispiele für Q-Anforderungen

- Q-Szenarien
  - Etwa 60 praxisnahe Qualitätsziele (aus konkreten Projekten/Systemen abstrahiert)
- Definitionen für typische Q-Anforderungen
  - 70+ häufig vorkommende Begriffe rund um Qualität – pragmatisch definiert

Beispiele für Q-Anforderungen

Beispiele für Qualitätsanforderungen an Software

Dr. Gernot Starke – gernot.starke@innoq.com – v0.6

Über Qualität

Die Qualität eines Produktes oder Systems ist ganz allgemein als Menge von Eigenschaften oder Merkmalen definiert.

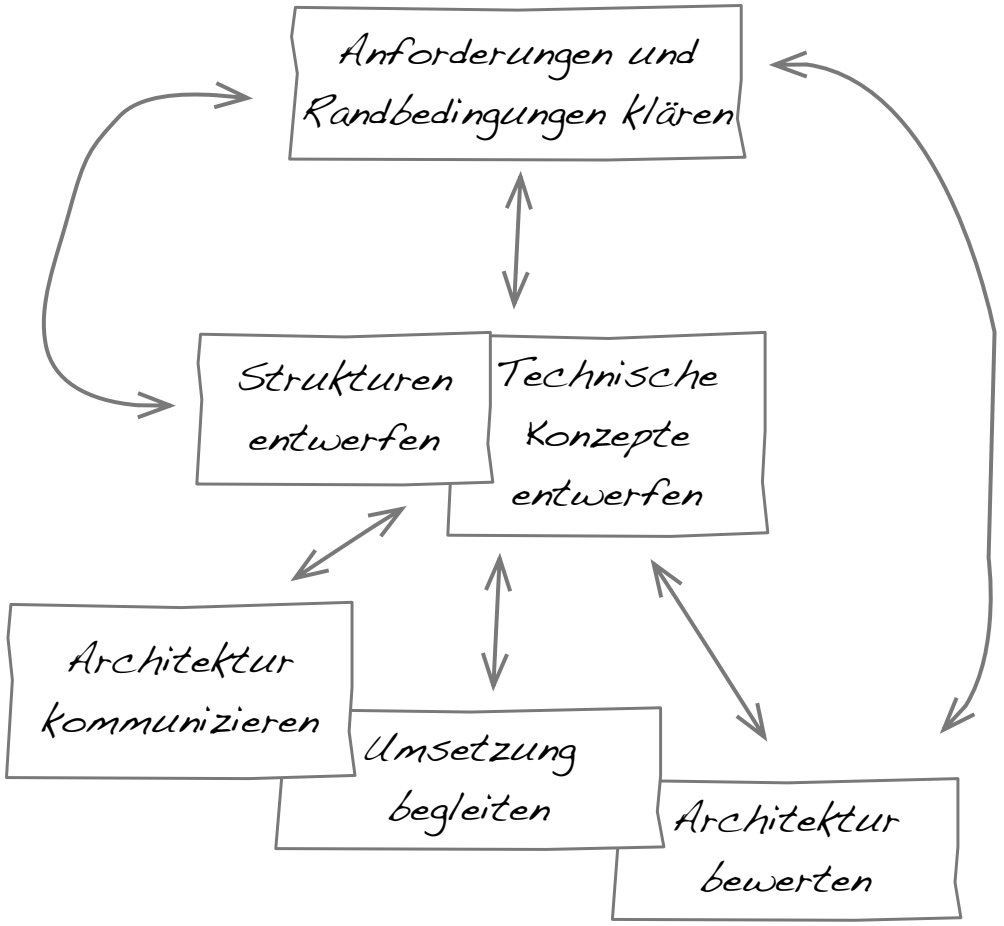
In der Praxis haben sich einige Kategorien (Oberbegriffe) für häufig auftretende *Qualitätsanforderungen* (synonym: Qualitätsziele) etabliert, im wesentlichen durch das DIN/ISO 9126 Begriffsmodell geprägt.

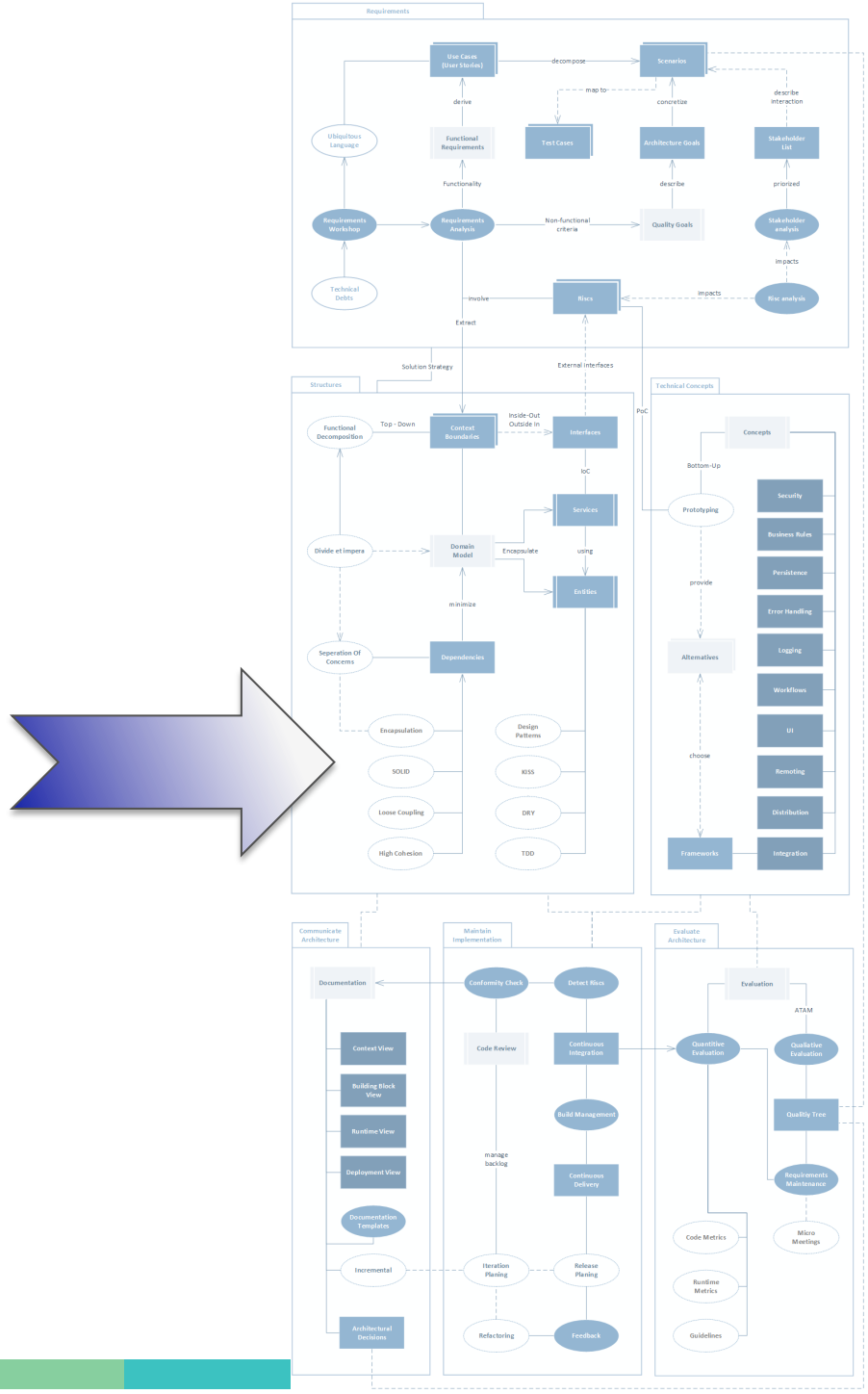
Die Kategorien

- Änderbarkeit
- Benutzbarkeit
- Effizienz
- Zuverlässigkeit
- Betreibbarkeit
- Sonstiges (u.a. Funktionalität)

Demo...

Architektur-Aufgaben...





https://github.com/arc42/arc-process

Dr. Gernot Starke

Gernot.Starke@innoQ.com

http://arc42.de
http://innoq.com